

# BÁO CÁO

Intern QC Engineer

Huỳnh Thị Yến Khoa

# MỤC LỤC

[1. Software testing (kiểm thử phần mềm) là gì? 1](#_Toc161653561)

[2. 7 nguyên tắc kiểm thử phần mềm 1](#_Toc161653562)

[3. Test Activities, Testware, Test role là gì? 2](#_Toc161653563)

[3.1 Test Activities: 2](#_Toc161653564)

[3.2 Testware: 4](#_Toc161653565)

[3.3 Test Roles: 4](#_Toc161653566)

[4. 4 level of testing 5](#_Toc161653567)

[4.1 Unit Testing (white box testing): 5](#_Toc161653568)

[4.2 Integration testing: 6](#_Toc161653569)

[4.3 System Test: 7](#_Toc161653570)

[4.4 Acceptance Test: 8](#_Toc161653571)

[5. Test Type 9](#_Toc161653572)

[5.1 Functional Testing: 9](#_Toc161653573)

[5.2 Non – funtional Testing: 11](#_Toc161653574)

[5.3 Structural Testing: 13](#_Toc161653575)

[5.4 Change-related Testing: 13](#_Toc161653576)

[6. API (Application Programming Interface) Testing 13](#_Toc161653577)

[6.1 Lợi ích của API Testing: 14](#_Toc161653578)

[6.2 Các công cụ API Testing 16](#_Toc161653579)

# NỘI DUNG

## Software testing (kiểm thử phần mềm) là gì?

* **Kiểm thử phần mềm (software testing):** là quá trình kiểm tra xem phần mềm có lỗi hay không, sản phẩm thực tế có đáp ứng được nhu cầu của người dùng bằng cách thủ công hoặc tự động (sử dụng các công cụ hỗ trợ Testing).
* **Mục đích kiểm thử:**

+ Tìm lỗi

+ Ngăn ngừa lỗi phát sinh

+ Đánh giá chất lượng của sản phẩm

+ Đảm bảo về chất lượng của sản phẩm đáp ứng nhu cầu của người dùng

## 7 nguyên tắc kiểm thử phần mềm

* **Testing shows the presence of defects (kiểm thử chứng minh sự hiện diện của lỗi):** Việc kiểm thử có thể chứng minh được phần mềm có lỗi nhưng không thể chứng minh phần mềm không có lỗi.
* **Exhaustive testing is impossible (kiểm thử toàn bộ là không thể):** các phần mềm, website hầu như đều thực thi trên nhiều nền tảng khác nhau. Vì vậy việc kiểm thử toàn bộ, toàn diện về một phần mềm là không thể. Vậy nên, thay vì tìm cách kiểm tra toàn bộ phần mềm, ta nên phần chia theo mức độ ưu tiên của từng lỗi trong phần mềm.
* **Early Testing (kiểm thử càng sớm càng tốt):** trong quy trình phát triển phần mềm, chúng ta cần phát hiện lỗi càng sớm càng tốt, cụ thể là ngay từ bước phân tích yêu cầu (Requirement) để tránh phát sinh chi phí về sau.
* **Defects Clustering (phân bố lỗi):** phần lớn lỗi thường tập trung vào các chức năng chính cùa phần mềm. Theo nguyên lý Pareto: 80% lỗi được tìm thấy trong 20% tính năng của hệ thống.
* **Pesticide Paradox (nghịch lý thuốc trừ sâu):** nếu sử đụng lặp đi lặp lại một bộ test case sẽ làm giảm khả năng tìm thấy lỗi vì phần mềm ngày càng phát triển và hoàn thiện hơn. Lỗi trước đó đã được xử lý theo test case cũ. Nên Tester cần kiểm tra và điều chỉnh Test Case thường xuyên đên phát hiện ra lỗi mới.
* **Testing is context dependent (kiểm thử phù thuộc vào ngữ cảnh):** tùy thuộc vào sản phầm là phần mềm, website, application sẽ có chiến lược kiểm thử khách nhau.
* **Absence of errors fallacy (quan niệm sai về lỗi):** phần mềm có thể 99% không có lỗi nhưng vẫn không sử dụng được vì sai yêu cầu của khách hàng.

## Test Activities, Testware, Test role là gì?

### Test Activities

* *Test planning:*

+ Task: Xác định mục tiêu của kiểm thử, xây dựng lịch trình kiểm thử để đáp ứng thời hạn của dự án.

+ Product: Test basis (SRS – Software Requirement Spectification, BRS – Business Requirement Spectification), tiêu chí đầu ra của sản phẩm.

* *Test monitoring and control:*

+ Task: so sánh quy trình thực tế với các test plan đã xây dựng, đánh giá chất lượng hệ thống dựa trên test result.

+ Product: Test progress report, test sumary report.

* *Test analysis:*

+ Task:

* Xác định ưu tiên các điều kiện thử nghiệm (đặc điểm chức năng, phi chức năng, các yếu tố kinh doanh và kĩ thuật).
* Báo cáo phân tích rủi ro.
* Xác định tính năng sẽ được kiểm tra.
* Xác định test conditions cho từng chức năng.

+ Product: tài liệu về test conditions

* Test design:

+ Task:

* Xây dựng Test Conditions thành các high level Test case.
* Tạo độ ưu tiên cho các test case.
* Xác định dữ liệu kiểm thử.
* Tạo môi trường kiếm thử, cơ sở hạ tầng và công cụ cần thiết cho việc kiểm thử.

+ Product: High level Test Case (không cần giá trị đầu vào và đầu ta mong muốn), môi trường kiểm thử, cơ sở hạ tầng và công cụ hỗ trợ việc kiểm thử. Cập nhật Test Condition nếu có phản hồi.

* Test implementaion:

+ Task:

* Xét độ ưu tiên của các Test procedures.
* Sắp xếp các bộ Test Suites theo kế hoạch kiểm thử sao cho đạt hiểu quả tốt nhất.
* Kiểm tra môi trường kiểm thử đã được thiết lập đầy đủ.

+ Product: Test Procedures, Test Suites, kế hoạch kiểm thử, môi trường kiểm thử, kết quả được mong đợi.

* Test execution:

+ Task:

* Thực thi các bộ Test Suites theo kế hoạch đã đưa ra trước đó.
* So sánh kết quả thực tế với kết quả mong đợi.
* Viết Bug Report.
* Lặp lại các hoạt động kiểm tra.

+ Product: Bug Report, Test Results, Defect Reports.

* Test completion:

+ task:

* Kiểm tra xem các lỗi đã được đóng hay chưa.
* Báo cáo tóm tắt hoạt động kiểm thử.
* Lưu trữ môi trường kiểm thử phần mềm để tải sử dụng.

+ Product:

* Báo cáo tóm tắt hoạt động kiểm thử.
* Phần mềm cuối cùng.

### Testware

Là đối tượng được tạo ra trong quá trình kiểm thử phần mềm.

Bao gồm:

* Test Cases: kịch bản, điều kiện các bước kiểm thử trong quá trình kiểm thử.
* Test Scripts: tập lệnh kiểm tra tự động thực thi bằng các công cụ kiểm thử.
* Test Data: dữ liệu sử dụng trong quá trình kiểm thử, có thể bao gồm dữ liệu đầu vào, kết quả mong đợi và cấu hình.
* Test Plans: Tài liệu ghi thông tin kế hoạch kiểm thử, mục tiêu, môi trường kiểm thử, công cụ sử dụng trong quá trình kiểm thử.
* Test Reports: kết quả kiểm thử, và số lỗi phát hiện được trong quá trình kiểm thử.
* Test Enviroments: thiết lập, cấu hình môi trường cho quá trình kiểm thử (bao gồm phần cứng và phần mềm).
* Test Tools: công cụ quản lý và theo dõi quá trình kiểm thử, sử dụng để tự động hóa việc kiểm thử.

### Test Roles

* trách nhiệm, vai trò của cá nhân hoặc nhóm trong quá trình kiểm thử.
* Test Manager: là người lập kế hoạch, quản lý các hoạt động kiểm thử. Đảm bảo dự án theo đúng như kế hoạch đã đưa ra.
* Test Analyst: thiết kế các Test Case và thực hiện các Test Case, xác định điều kiện kiểm thử (Test Conditions).
* Test Automation Engineer: viết và duy trì Test Scripts để đảm bảo tự động hóa trong quá trình kiểm thử.
* Defect manager: theo dõi, báo cáo và giải quyết lỗi.
* Test Enviroment Manager: cấu hình và triển khai môi trường phục vụ cho việc kiểm thử (bao gồm vả phần cứng và phần mềm).
* Security Tester: kiểm tra các lỗ hổng bảo mật, đảm bảo phần mềm được an toàn.

## 4 level of testing

### Unit Testing

Kiểm tra xem các Unit (funtion, Procedure, class, method) trong phần mềm có đáp ứng được chức năng hay không. Unit Testing là mức test nhỏ nhất. Được thực hiện càng sớm càng tốt và thực hiện xuyên suốt trong quá trình của dự án.

* **Mock Projects:** đối tượng ảo mô phỏng các tính chất và hành vi giống hệt như đối tượng thực được truyền vào bên trong khối mã đang vận hành nhằm kiểm tra tính đúng đắn của đối tượng đó.
* **Stub:** là một chương trình hoặc thành phần giả lặp thay thể cho chương trình hoặc thành phần chưa code xong, sử dụng trong trường hợp đến giai đoạn test mà vẫn chưa đủ các modules thì cần một thành phần là Stub đề giả lập nhiệm vụ của code còn thiếu để tiến hành quá trình test.

Lợi ích của Unit Test:

* Giúp tăng sự tin tưởng vào mã nguồn được thay đổi hoặc bảo trì.
* Có thể kiểm thử từng thành phần riêng lẻ của mã nguồn.
* Do thực hiện test trên từng đơn vị nhỏ của các modules riêng lẻ nên khi phát hiện lỗi cũng dễ khoanh vùng và sửa chữa.
* Có thể tái sử dụng mã nguồn, chuyển sang các dự án mới.
* Chi phí sửa chữa lỗi unit test thấp hơn ở các giai đoạn sau.

Nhược điểm của Unit Test:

* Unit test chỉ kiểm tra các tập dữ liệu và chức năng của nó chứ không kiểm tra được lỗi tích hợp.
* Có thể phải viết nhiều dòng code hơn để kiểm tra 1 dòng code.
* Cần đầu tư nhiều thời gian, công sức để học tập.

Công cụ hỗ trợ Unit Test



### Integration testing

Thực hiện kiểm thử tích hợp một nhóm Module có liên quan với nhau một cách hợp lý.

* **Mục đích:** phát hiện lỗi giữa các Unit, tập trung chủ yếu vào các giao diện và thông tin giữa các module, tích hợp nhiều Modules thành các hệ thống nhỏ.
* **Cách tiếp cận:**

+ BigBang Integration Testing: các module được tích hợp cùng một lúc sau đó tiến hành kiểm thử, được thực hiện khi nhóm kiểm thử nhận được đầy đủ modules của toàn hệ thống.

+ Top Down: kiểm tra từ trên xuống, kiểm tra các modules ở cấp cao nhất cho đến các modules ở đơn vị thấp hơn.

+ Bottom Up: ngược lại với top down, kiểm tra modules ở cấp thấp trước cho đến các modules ở đơn vị cao.

* Quy trình kiểm tra tích hợp:

+ Chuẩn bị kế hoạch kiểm tra tích hợp.

+ Thiết kế kịch bản thử nghiệm, trường hợp và tập lệnh.

+ Thực hiện các trường hợp kiểm tra sau đó là báo cáo lỗi.

+ Theo dõi và kiểm tra lại các lỗi.

+ Các bước 3 và 4 lặp lại cho đến khi hoàn tất tích hợp.

* Ví dụ về kiểm thử tích hợp:

+ Mục tiêu trường hợp kiểm thử: test liên kết giao diện giữa modules hộp thư và xóa thư.

+ Mô tả trường hợp kiểm thử: mở hộp thư, chọn ngẫu nhiên 1 email và click button xóa.

+ Kết quả mong đợi: email đã chọn xóa xuất hiện trong mục đã xóa/thùng rác.

### System Test

* Kiểm tra toàn bộ chức năng và giao diện của hệ thống.
* **Mục đích:** đánh giá hệ thống có đáp ứng đúng yêu cầu nghiệp vụ, chức năng đã đưa ra trước đó hay không.
* **Phân loại:**

*+ Kiểm thử chức năng (Funtional Test):* kiểm thử toàn bộ hệ thống, đảm bảo hệ thống hoạt động theo đúng yêu cầu.

*+ Kiểm thử khả năng bảo mật (Security Test):* kiểm tra hệ thống được bảo vệ an toàn, có nguy cơ bị đánh cắp dữ liệu hoặc có sự tấn công từ bên ngoài.

*+ Kiểm thử tính khả dụng (Usability Test):* kiểm tra tính thân thiện, dễ nhìn, dễ sử dụng của hệ thống.

*+ Kiểm tra tính tương thích (Compatibility Test):* kiểm tra hệ thống có tương thích với các yếu tố khác của hệ thống hay không?

*+ Kiểm tra khả năng phục hồi (Recovery Test):* kiểm tra khả năng khôi phục của hệ thống khi gặp các sự cố bất thường.

*+ Kiểm thử hồi quy (regression Testing):* kiểm tra xem có lỗi nào phát sinh sau khi đã sửa lỗi trước đó hay không.

*+ Migration Testing:* đảm bảo tính linh hoạt của hệ thống khi chuyển đổi sang cơ sở hạ tầng mới mà không gặp sự cố nào.

* Ưu điểm của System testing:

+ System test bao gồm kịch bản kiểm thử từ đầu tới cuối.

+ Thực hiện trong môi trường giống với môi trường Production, giúp hiểu được quan điểm của người dùng, tránh gặp issue khi hệ thống go live.

+ Test được cả kiến trúc ứng dụng và yêu cầu Business.

* Các bước thực hiện System Testing:

+ Tạo kế hoạch kiểm thử.

+ Tạo system test case và test script.

+ Chuẩn bị dữ liệu test.

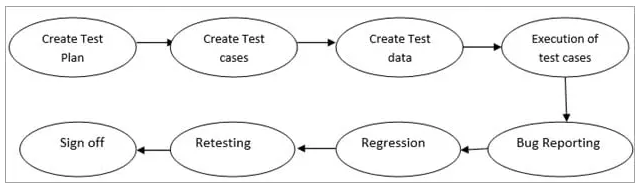
+ Thực hiện system test case và script.

+ Viết báo cáo bug, re-test sau khi đã fix.

+ Test hồi quy để xác minh ảnh hưởng của việc thay đổi code trước đó.

+ Lặp lại vòng đời kiểm thử cho đến khi hệ thống sẳn sàng.

+ Sign off khỏi đội test.



### Acceptance Test

* Kiểm tra xem phần mềm đã thỏa mãn yêu cầu của khách hàng hay chưa? Acceptance thuộc Black Box testing.
* *Mục đích:* kiểm tra lại hệ thống trước khi đưa vào hoạt động hoặc delivery sản phẩm cho khách hàng.
* *Phân loại:*

+ Alpha Test: sử dụng xác định sản phẩm trong môi trường kiểm thử được phát triển bởi một nhóm tester riêng biệt.

+ Beta Test: thực hiện mởi người dùng cuối cùng (thường là khách hàng). Thực hiện tại địa điểm của khách hàng.

+ Contract Acceptance testing: thực hiện để kiểm tra các tiêu chí và thông số kỹ thuật đã được xác định trong hợp đồng.

+ Regulation Acceptance testing: kiểm tra nhằm xác định phần mềm có tuân thủ các quy định không. Trong quá trình kiểm tra , Tester cần lưu ý tới những quy định của chính phủ và yêu cầu về mặt pháp lý.

+ Operational Acceptance testing: giúp đảm bảo các quy trình công việc cho phép các phần mềm hoặc hệ thống được sử dụng. Các bước kiểm thử này sẽ gồm: các quy trình công việc cho kế hoạch dự phòng quy trình đào tạo người dùng quy trình bảo trì và bảo mật.

+ User Acceptance testing: quá trình xác nhận phần mềm tạo ra có hoạt động phù hợp với người dùng cuối cùng hay không. Người thực hiện UAT là khách hàng hoặc người dùng cuối. Các bước thực hiện UAT gồm:

* Phân tích yêu cầu nghiệp vụ của khách hàng.
* Tạo kế hoạch kiểm tra UAT.
* Xác định Test Scenario.
* Chuẩn bị Data Test (giống với data thật nhất).
* Thực hiện kiểm thử.
* Ghi nhận kết quả.
* Xác nhận các chức năng của sản phẩm.

## Test Type

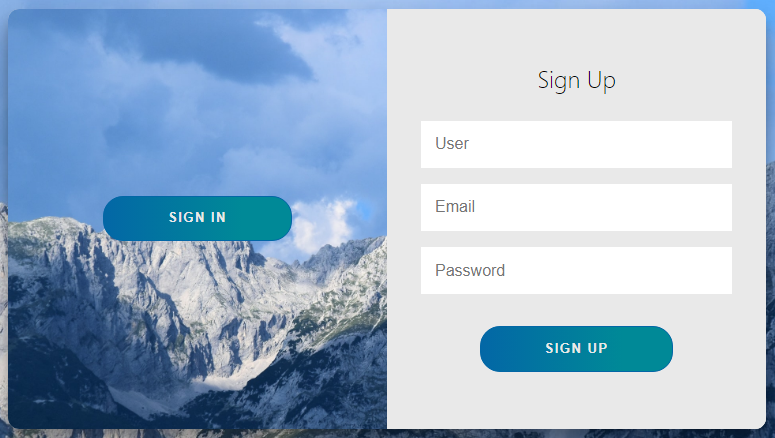
* Là sự phân loại các hoạt động kiểm thử theo mục đích, chiến thuật kiểm thử. Mỗi test type gắn với một mục tiêu kiểm thử nhất định.
* 4 loại Test type:

### Functional Testing

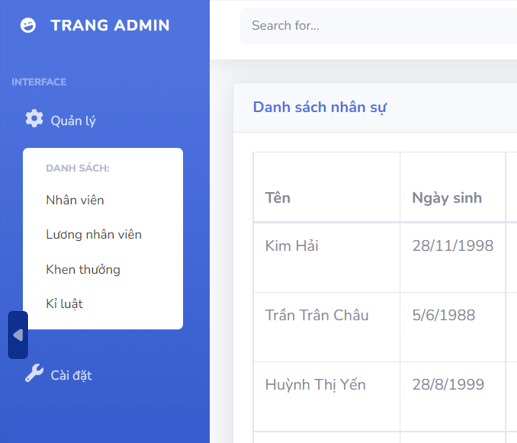
là một loại kiểm thử hộp đen (Black Box), kiểm tra xem phần mềm chức năng của hệ thống có chạy đúng theo yêu cầu hay không.

* Một số kĩ thuật kiểm thử phần mềm:
* Kiểm thử điều hướng người dùng: kiểm tra sự liên kết giữa màn hình này tới màn hình khác của hệ thống, đảm bảo cho sự hoạt động liên tục của nghiệp vụ đang thực hiện. Ví dụ như:

Hệ thống đăng nhập, đăng xuất



Thanh điều hướng



* Kiểm thử thao tác trên màn hình: thông thường sẽ là form nhập liệu, button chức năng, option lựa chọn. Với mỗi ô nhập liệu, các button, options, tester cần xác định các trường hợp thao tác trên màn hình theo yêu cầu đặt tả, tài liệu người dùng hay tài liệu quản trị viên tương ứng.
* Kiểm thử luồng thực hiện: thực hiện liền mạch một số thao tác qua nhiều bước với nhiều màn hình khác nhau để đánh giá xem có phù hợp với luồng nghiệp vụ không.
* Kiểm thử màn hình báo cáo: kiểm tra cách hiển thị hay tìm kiếm dữ liệu, dữ liệu hiển thị trong báo cáo có đúng hay không.

### Non – funtional Testing

kiểm thử những mặt phi chức năng của hệ thống như hiệu năng, khả năng sử dụng, bảo trì, ...

Kiểm thử phi chức năng gồm:

* Performance Testing: quá trình kiểm thử một phần hoặc cả hệ thống nhằm xác định độ ổn định, tốc độ load, khả năng mở rộng và độ đáp ứng của hệ thống theo lượng công việc nhất định.

Mục tiêu của kiểm thử hiệu năng là đánh giá:

* Tốc độ xử lý: xem hệ thống có phản hồi nhanh với số Request đã gửi lên hay không.
* Độ ổn định: xác định xem hệ thống có ổn định dưới các mức tải khác nhau hay không.
* Khả năng mở rộng: xác định số lượng người dùng tối đa mà hệ thống có thể xử lý được.
* Load Testing: kiểm tra hệ thống bằng cách tăng tải liên tục cho đến khi đạt mức tối đa. Là tập hợp con của Performance Testing. Các thuộc tính cần theo dõi trong Load Testing gồm: hiệu suất cao nhất, thông lượng máy chủ, thời gian đáp ứng các mức tải khác nhau (dưới ngưỡng ngắt), hệ thống xử lý được bào nhiêu người dùng có thể xử lý mà không ảnh hưởng tới hiệu suất.
* Stress testing: đo lường độ bền của hệ thống và khả năng xử lý lỗi trong điều kiện tài cực nặng, đảm bảo hệ thống không gặp sự cố trong các tình huống khủng hoảng.
* Usability Testing: đánh giá tính dễ sử dụng đối với người dùng về giao diện. Mục tiêu của Usability Testing là đáp ứng nhu cầu của người dùng, tập trung chủ yếu vào các thông số: hiệu quả của hệ thống, độ chính xác, người dùng thân thiện, ...
* Lợi ích của Usability Testing:
* Giảm chi phí thiết kế và phát triển, giúp tăng sự hài lòng của người dùng.
* Đẩy mạnh tính kinh doanh, mang lại sự hài lòng cho khách hàng.
* Phần mềm trở nên dễ hiểu, dễ sử dụng hơn với người dùng hoặc khách hàng.
* Tăng số lượng người sử dụng phần mềm.
* Thời gian làm quen với phần mềm của người dùng được rút ngắn.
* Maintainability Testing: quá trình kiểm thử xác định khả năng bảo trì của hệ thống.
* Reliability Testing: quá trình kiểm thử kiểm tra phần mềm có thể hoạt động mà không có lỗi trên một khoảng thời gian và một môi trường cụ thể. Mục đích của Reliability Testing là để kiểm tra xem phần mềm có đáp ứng được yêu cầu về độ tin cậy của khách hàng hay không.

Các loại Reliability Testing:

**Features Testing**: kiểm tra tính năng được cung cấp bởi phần mềm và được tiến hành theo các bước:

* + - Mỗi thao tác trong ứng dụng được thực hiện ít nhất một lần.
    - Giảm tương tác giữa 2 hoạt động.
    - Mỗi hoạt động phải chắc chắn rằng chúng thực hiện đúng.

**Load Testing**: khi phần mềm được đưa vào hoạt động, sau một thời gian sẽ xuống cấp, Load Testing được tiến hành để kiểm tra hiệu suất của phần mềm dưới mức tải tối đa.

**Regression Testing**: kiểm tra xem có lỗi nào phát sinh sau khi đã sửa lỗi trước đó hay không.

* Portability Testing: Test phần mềm với mục đích đảm bảo có thể tái sử dụng các chức năng và có thể chuyển sang phần mềm khác. Portability Tesing qua tâm đến việc chạy phần mềm trên các môi trường khác nhau.

### Structural Testing

Được coi là một loại White Box Testing, tập trung vào kiểm thử bên trong phần mềm hơn là kiểm thử chức năng, áp mục được ở mọi mức độ kiểm thử. Developer có thể ứng dụng Strucural Testing trong quá trình kiểm thử thành phần (unit tesing).

### Change-related Testing

Mục đích để kiểm tra xem hệ thống có vận hành tốt sau những lần sửa lỗi hay không. Gồm 2 loại chính:

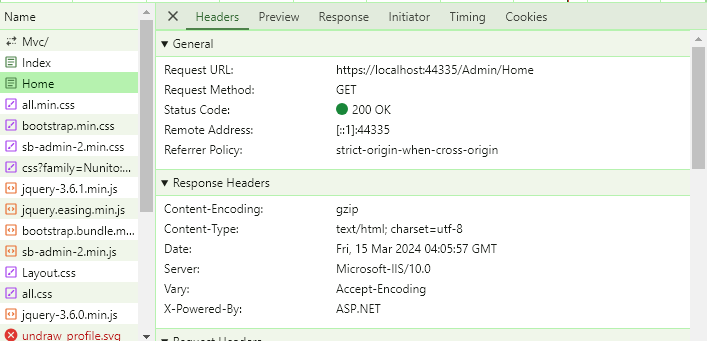
* Confirmation Testing: được thực hiện sau khi lỗi trong phần mềm được xác nhậm và sửa chữa. Vai trò là để xác nhận xem lỗi đã thực sự được sửa hay chưa. Được tiến hành bằng cách cho một input giống hệt ban đầu và xem output có như mong đợi hay không.
* Regression Testing: xác nhận rằng những thay đổi trong phần mềm không gây ảnh hưởng đến hệ thống mà vẫn đáp ứng những yêu cầu khác.

## API (Application Programming Interface) Testing

* API: là phương thức kết nối giữa Client và Server.
* API Testing: là loại kiểm thử giao diện lập trình ứng dụng một cách trực tiếp. API Testing là một phần của kiểm thử tích hợp kiểm tra xem hệ thống có đáp ứng những yêu cầu về tính năng, độ tin cậy, hiệu suất và bảo mật.
* API Testing được thực hiện ở tầng nghiệp vụ (Bussiness layer).
* Trong quá trình kiểm thử API, dữ liệu đuộc trao đổi từ XML hoặc JSON thông qua các yêu cầu phản hồi HTTP (HTTP requests and responses).
* API được xây dựng dựa trên 2 thành phần: Request và Reponse.

+ 1 request có 4 phương thức chính:

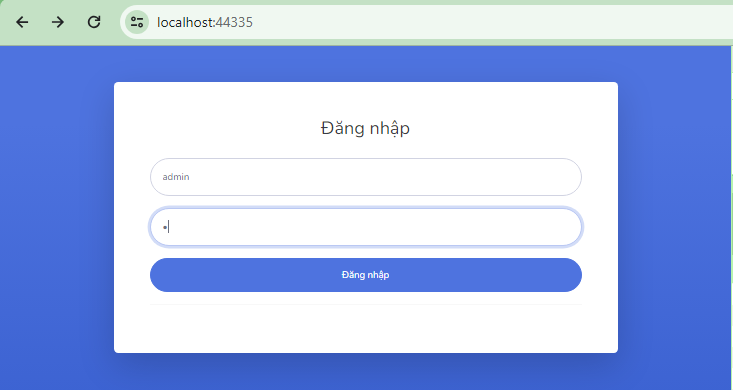
* Get: client sử dụng Get để truy cập các tài nguyên tại URL được chỉ định trên máy chủ. Có thể hiểu là lấy dữ liệu từ điều kiện đã được chỉ định.

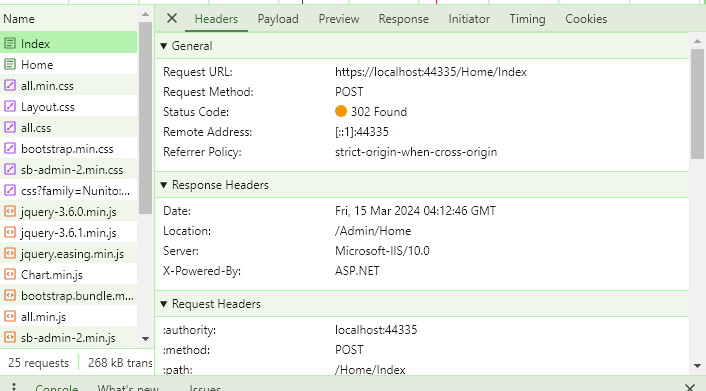


Như hình trên client gửi request lên với phương thức là Get để lấy dữ liệu tại trang home.

* Post: Client sử dụng Post để gửi dữ liệu đến máy chủ.

Ví dụ: client gửi yêu cầu đăng nhập lên máy chủ





* Put: Client sử dụng Put để thay thế hoặc cập nhật dữ liệu có trên máy chủ.
* Delete: client sử dung yêu cầu Delete để xóa tài nguyên có trên máy chủ.
* Ngoài ra còn có các phương thức khác:
* Head: giống với Get nhưng Reponse trả về không có Body chỉ có Header.
* Patch: ghi đè thông thay được thay đổi của đối tượng.
* Connect: thiết lập một kết nối tới Server theo URL.
* Options: mô tả các tùy chọn giao tiếp cho Resource.
* Trace: thực hiện một bài test Loop-back theo đường dẫn đến Resource.

### Lợi ích của API Testing

* Đảm bảo tính đúng đắn.
* Đảm bảo ổn định.
* Tăng độ tin cậy và chất lượng.
* Giảm rủi ro và chi phí.

### Nhược điểm của API Testing

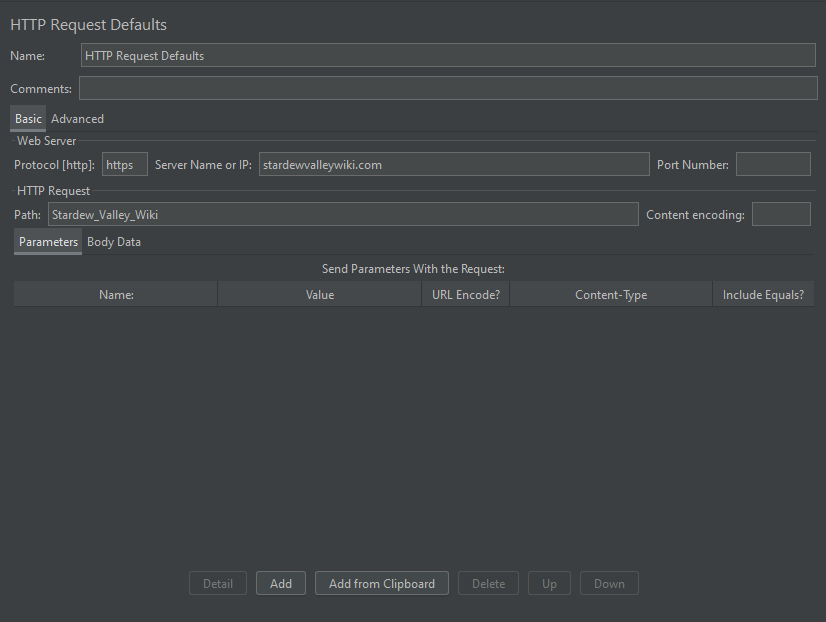
* Để sử dụng hiệu quả cần có kiến thức chuyên sâu, có kinh nghiệm backend tốt.
* Phát triển, nâng cấp và cận hành API là một quá trình lâu dài, tốn thời gian và chi phí.
* Nếu hệ thống vị tấn công trong khi chủ sở hữu chưa giới hạn điều kiện kĩ, sẽ rất khó để bảo mật hệ thống.

### Các công cụ API Testing

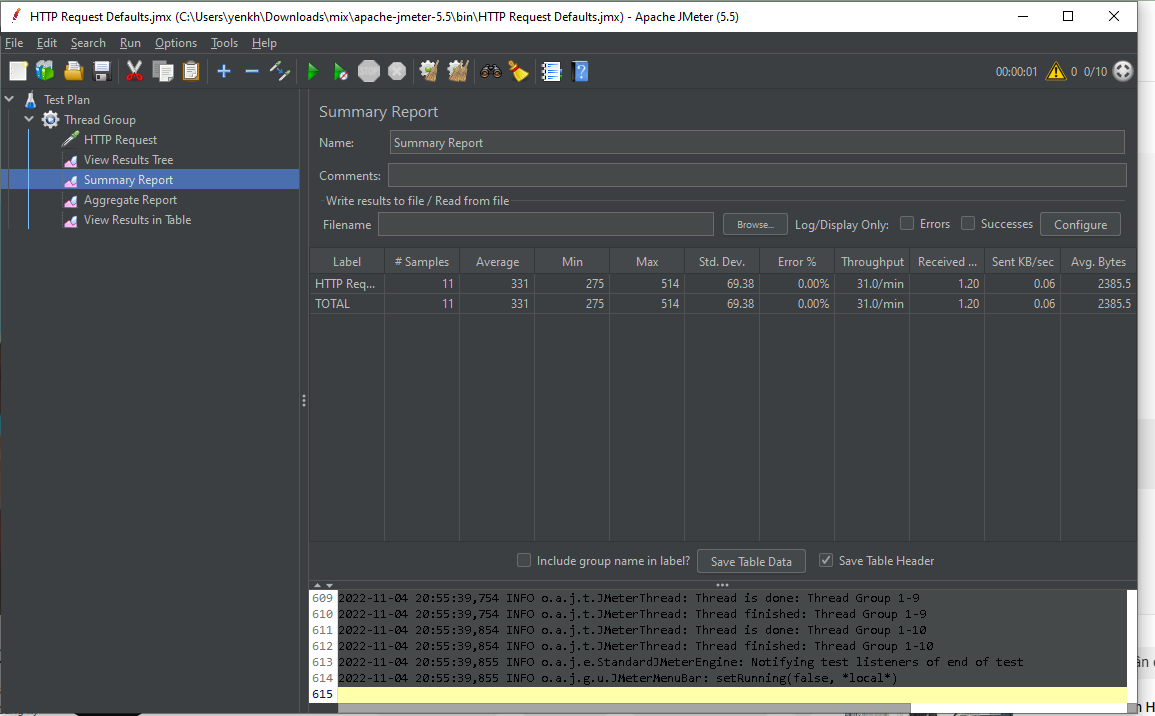


* Jmeter:

+ HTTP Request trong Jmeter

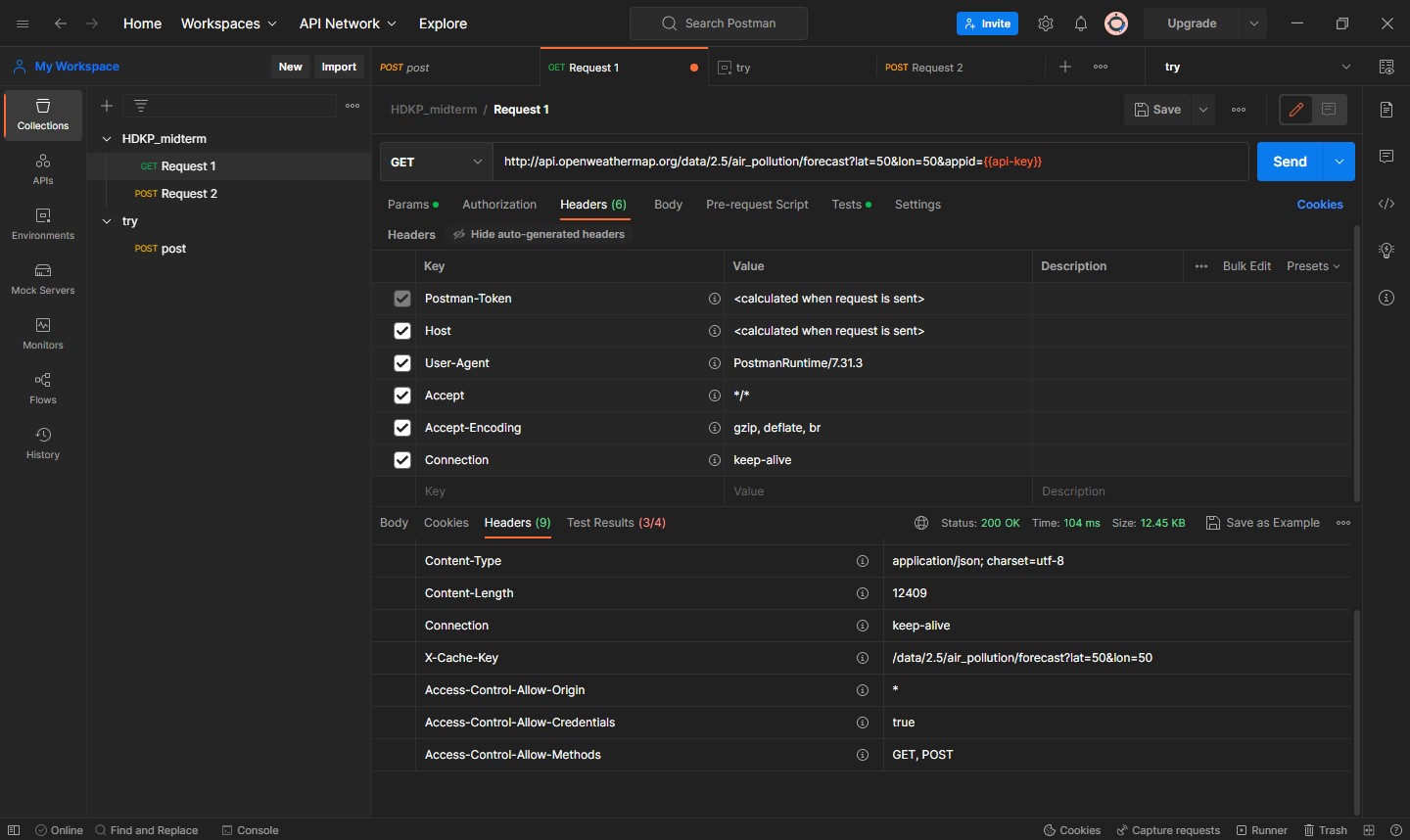


+ Sumary Report gồm: Samples (tổng số lần run), Average (trung bình cộng thời gian phản hồi của các lần run), min và max (thời gian phản hồi nhỏ nhất và lớn nhất của các request), Std.Dev (độ lệch chuẩn, Std.Dev càng thấp thì hiệu năng càng tốt, độ lệch chuẩn nên nhỏ hơn giá trị trung bình), error (số % những user bị fail), throughput (số lượng request mà server xử lý được trong 1 đơn vị thời gian nhất định), received KB/s và sent KB/s (lưu lượng server trả về và lưu lượng client gửi lên server), Avg.bytes (lưu lượng trung bình mà server trả về).

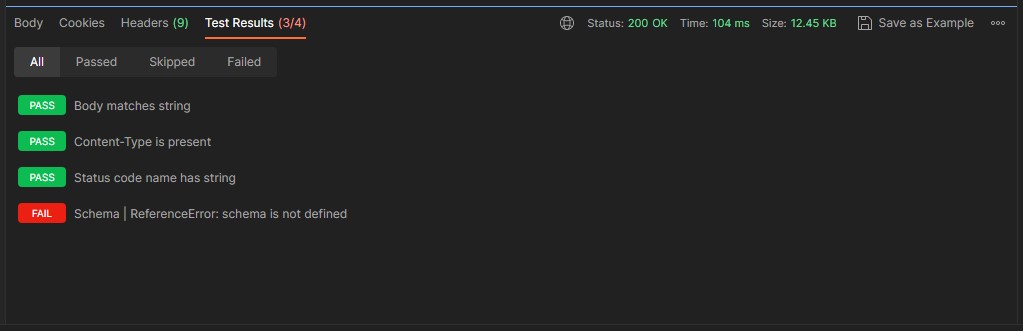


* Postman:

Ví dụ: gửi Request với phương thức GET



Test result:



Test scripts mẫu có trong Postman

